ГЛАВА XX.
Метрические алгоритмы

Близость – это не устранение дистанции,
а ее преодоление.
Д. Глаттауэр

Нам не дано оказаться бесконечно
близко к тем, кого мы любим.
А. С. Пиньоль

Описанные в этой главе алгоритмы используют информацию о попарных расстояниях между объектами и называются **метрическими**, в англоязычной литературе их называют «distance-based», а также «memory-based», «instance-based», «non-parametric». В задаче обучения по размеченной выборке при определении метки нового объекта x они анализируют расстояния

$$\rho(x,x_1),\ldots,\rho(x,x_m),$$

до объектов обучающей выборки, т.е. во всей главе предполагается, что в пространстве объектов задана метрика ρ, которая правильно отражает семантику похожести объектов (ниже поговорим про выбор метрики). Два основных представителя этого семейства алгоритмов:

- Ближайший центроид (Nearest Centroid Algorithm / Distance from Means),
- метод k ближайших соседей (kNN, k Nearest Neighbors[1]).

Результаты работы этих алгоритмов мы будем иллюстрировать на модельных задачах классификации, см. рис. XX.1: «две кучки» и «вложенные полумесяцы». Отметим, что кроме функции расстояния в описываемых алгоритмах можно использовать и функцию сходства, например косинусную меру сходства $\cos(x,z)$ (производя естественные замены в описаниях алгоритмов).

[1] T.M. Cover, P.E. Hart «Nearest Neighbor Pattern Classification», IEEE Trans. Inform. Theory, Vol. IT-13, pp 21-27, 1967.

Ближайший центроид (Nearest centroid algorithm)

Рассмотрим задачу классификации на непересекающиеся классы с вещественными признаками, в ней $Y = \{1, 2, \ldots, l\}$, $x_i \in \mathbb{R}^n$. **Центроидами** классов называются центры масс объектов классов из обучающей выборки:

$$c_j = \frac{1}{|\{i : y_i = j\}|} \sum_{i: y_i = j} x_i$$

(здесь для j-го класса усреднили все признаковые описания объектов с меткой «j»), $j \in Y$. Классификация методом ближайшего центроида задаётся формулой:

$$a(x) = \arg\min_j \rho(x, c_j),$$

т.е. объект относится к тому классу, к центроиду которого он ближе. На рис. XX.1 показаны разделяющие поверхности этого метода в наших модельных задачах (для двух классов и эвклидовой метрики это всегда гиперплоскость).

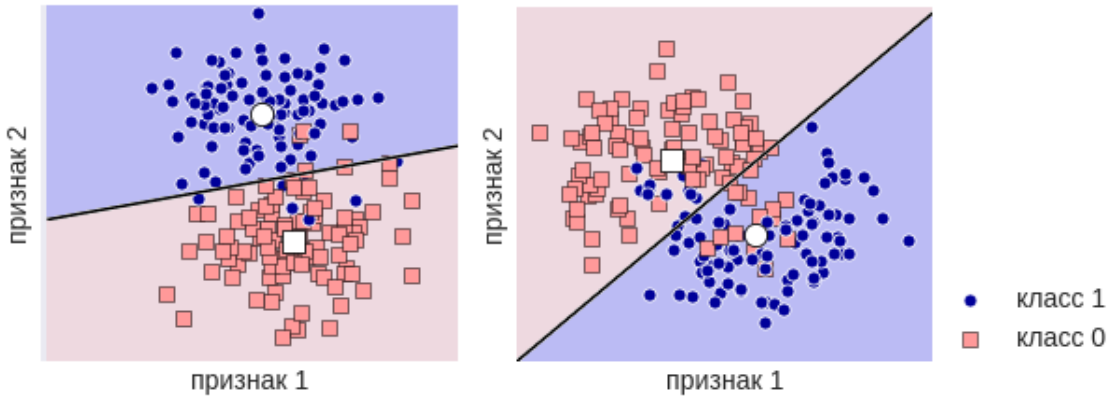

Рис. XX.1. Разделяющие поверхности метода ближайшего центроида.

Особенности алгоритма «Ближайший центроид»:

- процедура обучения состоит в вычислении центроидов,
- после обучения достаточно хранить только центроиды (их можно адаптивно менять при пополнении выборки), поэтому

  размер модели = число классов × размер описания центроида,

- понятие центроида можно менять (допустима любая формализация понятия «средний объект», см. главу о средних),

- у метода простая реализация на современных языках программирования,
- очень простой алгоритм в смысле вида разделяющей поверхности (интуитивно подходит в задачах, где объекты разных классов распределены «колоколообразно»).

При желании можно обобщить метод и описывать класс не одним вектором (центроидом), а несколькими. Например, такие векторы можно получить с помощью кластеризации объектов каждого класса (см. главу «Кластеризация»).

Ближайший центроид – максимально сжатое описание классов.

Подход, основанный на близости и окрестности

В подходе, основанном на близости, задачи машинного обучения решаются по следующей логике: вводится понятие **окрестности (neighborhood)** объекта x – $N(x)$ (похожие на него объекты, см. рис. XX.2), алгоритм a вычисляет метки по формулам

Одна из центральных идей в машинном обучении – исследовать похожие объекты.

$$a(x) = \text{mode}(y_i \mid x_i \in N(x)) \text{ – для задачи классификации,}$$

(т.е. определяет самую популярную метку объектов обучающей выборки, которые попали в окрестность классифицируемого объекта),

$$a(x) = \text{mean}(y_i \mid x_i \in N(x)) \text{ – для задача регрессии,}$$

(т.е. усредняет метки объектов обучающей выборки, которые попали в окрестность).

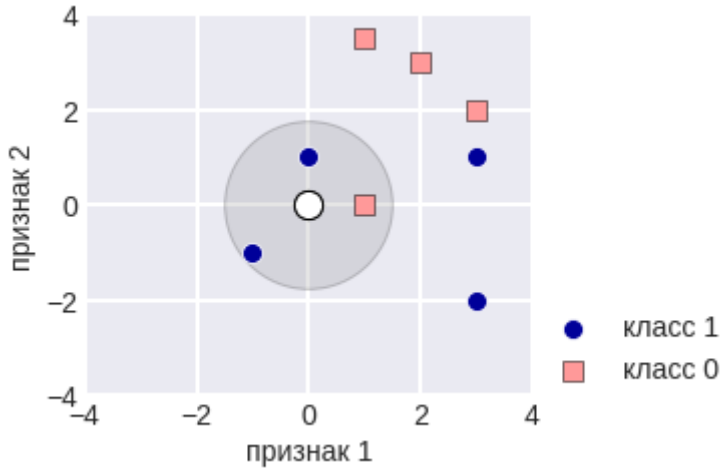

Рис. XX.2. Окрестность объекта.

Теперь опишем, как выбирается окрестность. Обычно в математике окрестность задаётся своим радиусом $R \in \mathbb{R}^+$:

$$N(x) = \{x_t \mid \rho(x_t, x) \le R\},$$

при такой формализации получаем **метод ближайшего соседа с фиксированным радиусом (Fixed-Radius Near Neighbor)**, который, однако, не слишком популярен.

Предложим другой способ формализации понятия «окрестность». Если X – метрическое пространство с метрикой ρ, нумерация объектов такая, что

$$\rho(x, x_1) \le \ldots \le \rho(x, x_m)$$

(мы упорядочили объекты по удалению от объекта x), то окрестностью объекта x можно считать множество его k наиболее близких обучающих объектов:

$$N(x) = \{x_1, \ldots, x_k\},$$

которые и называются **соседями**. Так определяется **метод *k* ближайших соседей** (kNN, k Nearest Neighbors). На см. рис. XX.3 показаны окрестности (ближайшие k объектов – они обведены окружностью) при разных k. При $k = 1$ алгоритм классификации называется **алгоритмом ближайшего соседа (Nearest Neighbor Algorithm).**

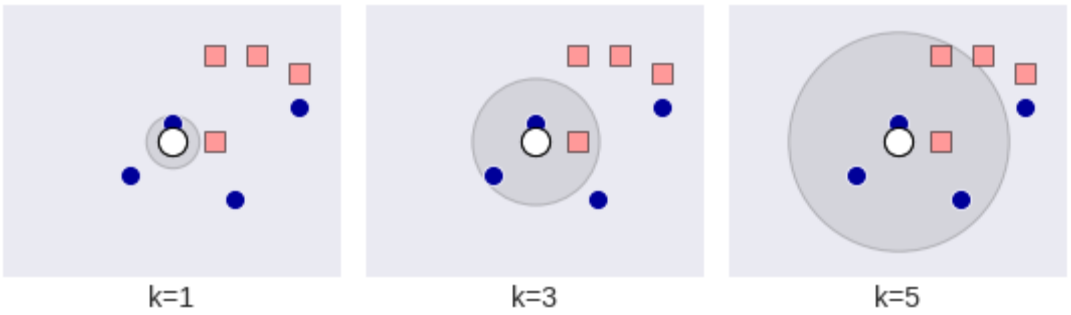

Рис. XX.3. Разные окрестности в kNN в зависимости от k.

Отметим особенности работы kNN:

- формально нет обучения – храним всю обучающую выборку[1],

[1] Дальше увидим, что при обучении выборка может сохраняться в специальной структуре данных, что позволит впоследствии быстрее находить ближайших соседей.

- при определении метки (работе алгоритма) – просматриваем всю выборку (и вычисляем расстояния до каждого объекта обучения).

Алгоритмы с такими свойствами называются **ленивыми алгоритмами** (lazy learners), в противоположность **нетерпиливым алгоритмам** (eager learners), которые по обучающей выборке получают значения параметров модели, а при работе не используют обучающую выборку, а лишь полученные значения. Примером нетерпеливого алгоритма является изученный выше «Ближайший центроид».

Гиперпараметр k (число соседей) можно выбрать по качеству на отложенной выборке (или на скользящем контроле – см. главу «Выбор модели»), также отметим, что формально гиперпараметром алгоритма является метрика[1]. Приведём график качества (долю правильных ответов) на обучающей выборке и отложенной тестовой выборке от числа соседей (n_neighbors[2]), см. рис. XX.4. Здесь использовалась вторая модельная задача с внесённым дисбалансом классов (объектов класса 1 было в два раза меньше), поскольку в этом случае большие значения k заведомо невыгодны. Видно, что на тестовой выборке оптимальное число соседей $k=7$ (схожее качество наблюдается при нечётных k от 7 до 13). Также заметны «скачки» качества при изменении чётности числа соседей (в задаче с двумя классами при чётном k возможны ничьи: половина объектов принадлежат одному классу, другая половина – другому, которые разрешаются произвольно[3], из-за чего непредсказуемо меняется качество).

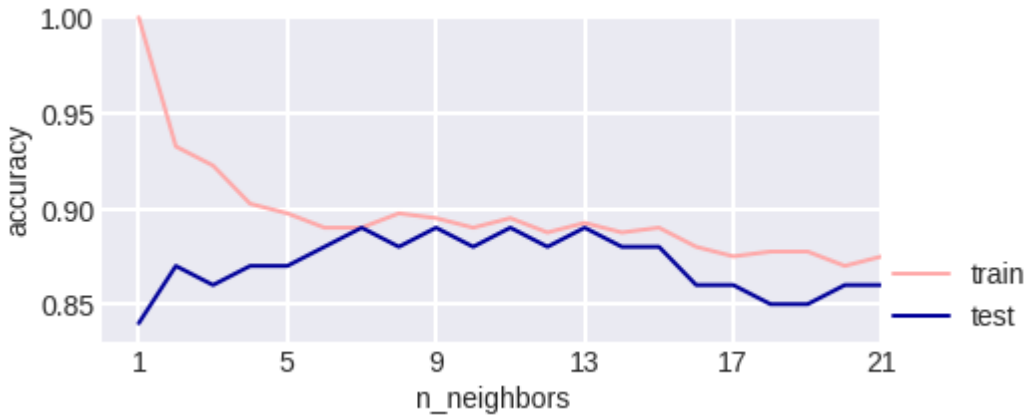

Рис. XX.4. Графики качества метода kNN при изменении числа соседей k.

[1] Или параметры метрики или ядро (см. дальше).

[2] Так соответствующий гиперпараметр называется в sklearn.

[3] Зависит от реализации, чаще объект относят к классу с наименьшим номером, т.к. классификация производится с помощью функции argmax.

Отметим, что при $k=1$ качество на обучающей выборке метода kNN максимально, т.к. каждый объект близок сам к себе[1].

Есть даже теоретические гарантии надёжности метода ближайшего соседа, чтобы их лучше понять следует изучить главу про байесовские алгоритмы, тогда будет понятно, что такое «оптимальный алгоритм». Отметим, что это не алгоритм, который всегда выдаёт правильный ответ – такое невозможно в общем случае. Это алгоритм, который при известных (откуда-то) вероятностях принадлежности к классам объекта выдаёт в качестве ответа класс с максимальной вероятностью.

Теорема. В достаточно однородном метрическом пространстве[2] объектов бинарной задачи классификации ошибка 1NN не выше удвоенной ошибки оптимального алгоритма.

Доказательство. Пусть в некоторой области p – вероятность встретить объект класса 0 (взяли этот класс для определённости), пусть также $0 \le p \le 0.5$ (при $p > 0.5$ проводим рассуждения для класса 1). «Достаточную однородность» нашего пространства понимаем в том смысле, что ближайший объект из обучающей выборки лежит в той же области и имеет ту же вероятность принадлежать классу 0. Тогда p также и вероятность ошибки оптимального алгоритма, можно выписать такую таблицу, в которой показаны вероятности принадлежности объекта из области и его соседа классам.

	объект	
сосед	класс 1 – p	класс 0 – $(1-p)$
класс 1 – p	pp	$p(1-p)$
класс 0 – $(1-p)$	$p(1-p)$	$(1-p)(1-p)$

Вероятность ошибочной классификации объекта методом 1NN (когда классы объекта и его соседа не совпадают) равна сумме недиагональных элементов таблицы:

$$2p(1-p) = 2p - 2p^2 \le 2p,$$

[1] Ошибки могут быть, если объекты с одинаковым описанием имеют разные метки в обучающей выборке.

[2] Этот термин поясняется при доказательстве.

т.е. сверху ограничена удвоенной вероятностью ошибки оптимального алгоритма. **Теорема доказана.**

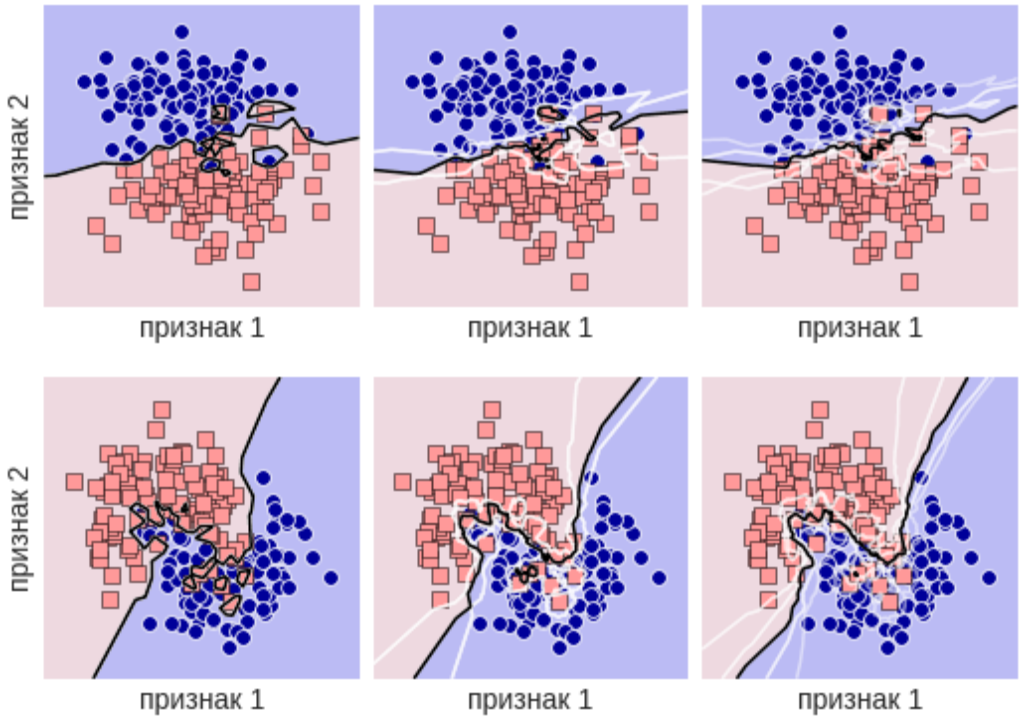

Рис. XX.5. Разделяющая поверхность метода kNN при числе соседей = 1, 3 и 5 в двух модельных задачах.

На рис. XX.5 показано решение модельных задач методом kNN при разном числе соседей: в первой строке – задача «две кучки», во второй – «два полумесяца», по столбцам – разные k: 1, 3, 5. Как увидим дальше, параметр k отвечает за «сложность модели», причём чем больше k, тем метод проще. Пока отметим, что при малых k совершаются достаточно очевидные ошибки: если среди объектов класса случайно оказался объект другого класса (его называют «выброс»), то объекты близкие к нему будут неверно классифицированы (на рис. XX.5 это видно, например, из наличия розовых островов в голубом море – области, в которой классификатор относит объекты к синему классу 1). При увеличении k такие ошибки исчезают, но при очень больших k (например, при $k = m$, где m – объём обучающей выборки) классификатор kNN вырождается: превращается в константный алгоритм и относит все объекты к большему классу (представителей которого больше в обучающей выборке).

Белые линии на рис. XX.5 делят плоскость на области одинаковой уверенности в ответе. При $k = 1$ их нет: каждая точка плоскости ближе к синей или розовой из обучения. При $k = 3$ у точки могут быть все соседи синие, 2а синих соседа из 3х (относим в синий класс), 1н синий сосед из 3х или все розовые соседи

(относим в розовый класс). Поэтому области четыре, они соответствуют оценкам вероятности принадлежности классу 1 вида:

$$i / k,\ i \in \{0,1,2,\ldots,k\}.$$

При $k = 5$ таких областей шесть.

Метод k ближайших соседей легко переносится на решение задачи регрессии, см. рис. XX.6: для объекта находим k ближайших из обучающей выборки, их метки усредняются.

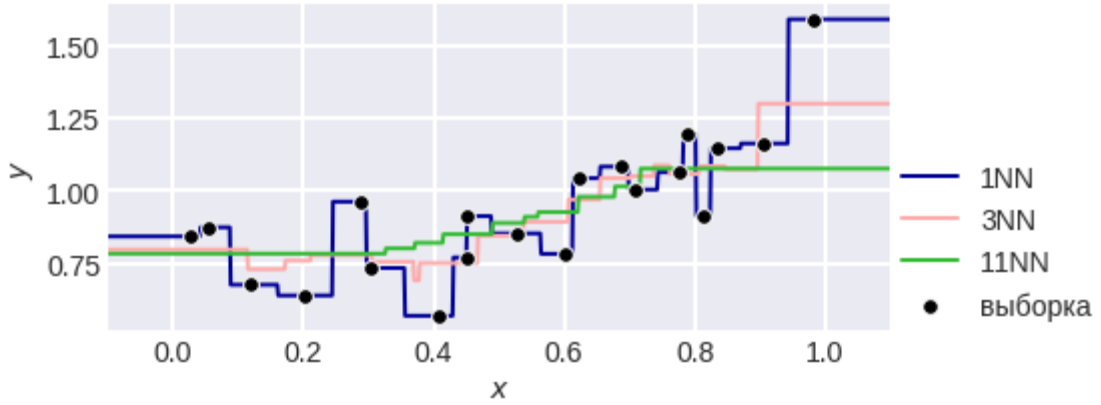

Рис. XX.6. Регрессионные кривые, построенные методом kNN при разных k.

Для kNN-регрессора также есть теоретические обоснования[1], например, доказывается свойство универсальной согласованности (universally consistent):

$$\mathbf{E}\,|a(x) - f(x)| \xrightarrow[n\to+\infty]{} 0$$

(т.е. решение сходится к истинному) в задаче регрессии при $y(x) = f(x) + \varepsilon$, ε – шум, $k \to +\infty$, $k / n \to 0$ и не слишком сильных вероятностных предположениях, функция f лежит в достаточном обширном классе функций.

Возможная проблема метода kNN показана на рис. XX.7. Здесь метод 5NN отнесёт объект в класс 0, хотя два самых близких объекта принадлежат классу 1, а остальные значительно удалены. Очевидно, что близкие соседи должны быть важнее!

Фундаментальный принцип ML: «более похожие объекты важнее».

[1] См. например G. H. Chen, D. Shah « Explaining the Success of Nearest Neighbor Methods in Prediction» // Publisher: Now Foundations and Trends https://devavrat.mit.edu/wp-content/uploads/2018/03/nn_survey.pdf

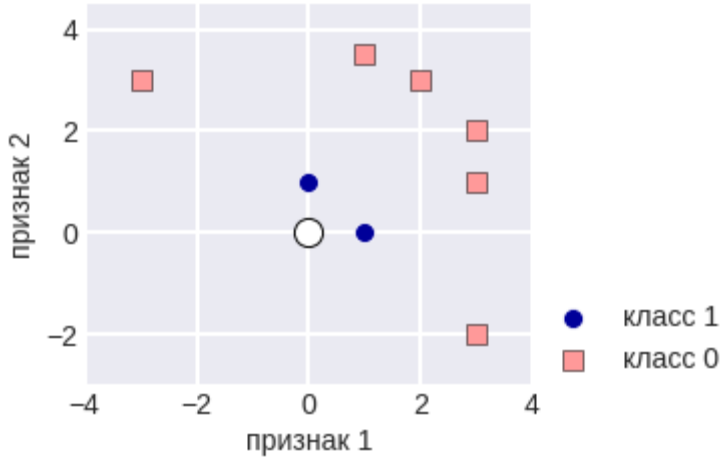

Рис. XX.7. Возможная точечная конфигурация в задаче бинарной классификации.

Для решения этой проблемы используются **весовые обобщения *k*NN**. Для этого представим классический алгоритм в виде суперпозиции[1] алгоритма $b: X \to \mathbb{R}^l$ вычисления оценок принадлежностей к классам:

$$b(x) = (b_1(x), \ldots, b_l(x)) \in \mathbb{R}^l$$

и решающего правила $c: \mathbb{R}^l \to \{1, 2, \ldots, l\}$, которое по полученным оценкам определяет номер класса. Интересно, что алгоритмы многих моделей можно представить в виде такой суперпозиции[2]. Определение kNN можно переписать в виде

$$a(x) \equiv \text{mode}(y_i \mid x_i \in N(x)) = \arg\max_j \sum_{t=1}^{k} I[y(x_t) = j],$$

(сумма индикаторов $I[\cdot = j]$ здесь совпадает с числом соседей из окрестности с меткой j), поэтому алгоритм вычисления оценок просто считает для каждого класса число объектов этого класса в окрестности или долю (иногда удобно пронормировать):

$$b_j(x) = \frac{\sum_{t=1}^{k} I[y(x_t) = j]}{k}, \qquad \text{(XX.1)}$$

а решающее правило просто выбирает наиболее вероятный класс:

[1] Последовательного применения.

[2] Есть даже теорема о таком представлении: Журавлев Ю. И. Об алгебраическом подходе к решению задач распознавания или классификации //Проблемы кибернетики. – 1978. – Т. 33. – С. 5-68.

$$a(x) = \arg\max_j b_j(x),$$

ведь полученные доли являются оценками принадлежности к классам. На рис. XX.5 белыми линиями как раз были показаны области с разным значением $b_1(x)$. Теперь обобщим формулу для вычисления оценок:

Стандартная схема: оцениваем вероятности принадлежности к классам, выбираем наиболее вероятный класс.

$$b_j(x) = \frac{\sum_{t=1}^{k} w_t I[y(x_t) = j]}{\sum_{t=1}^{k} w_t} \qquad \text{(XX.2)}$$

(если сумма весов равна 1, то нормировка не нужна), где веса у более близких объектов не ниже, чем у более далёких:

Переход к сумме индикаторов позволяет обобщить формулу с помощью весов объектов.

$$w_1 \geq w_2 \geq \ldots \geq w_k > 0.$$

При этом используются разные весовые схемы:

$$w_t = (k - t + 1)^{\delta}, \qquad \text{(XX.3)}$$

$$w_t = \frac{1}{t^{\delta}}, \qquad \text{(XX.4)}$$

$$w_t = K\left(\frac{\rho(x, x_t)}{h(x)}\right), \qquad \text{(XX.5)}$$

их значения показаны на рис. XX.8 при $\delta = 1$, $K(z) = \exp(-0.15z^2)$, веса также были пронормированы (чтобы их сумма была равной единице), расстояния до соседей в (XX.5) были 1, 2, ..., 5. Подробнее о весовых схемах см. главу «Весовые схемы». Логика первой схемы такая, если у нас есть k соседей, то пусть у них будут следующие веса:

$$k > (k-1) > \ldots > 1,$$

это линейная схема (см. рис. XX.8, слева), внести нелинейность можно с помощью параметра $\delta \in [0, +\infty)$:

$$k^{\delta} \geq (k-1)^{\delta} \geq \ldots \geq 1^{\delta}.$$

При $\delta = 0$ получаем константную схему, при больших δ – веса самых ближних объектов существенно больше остальных. Аналогично можно объяснить и другие весовые схемы.

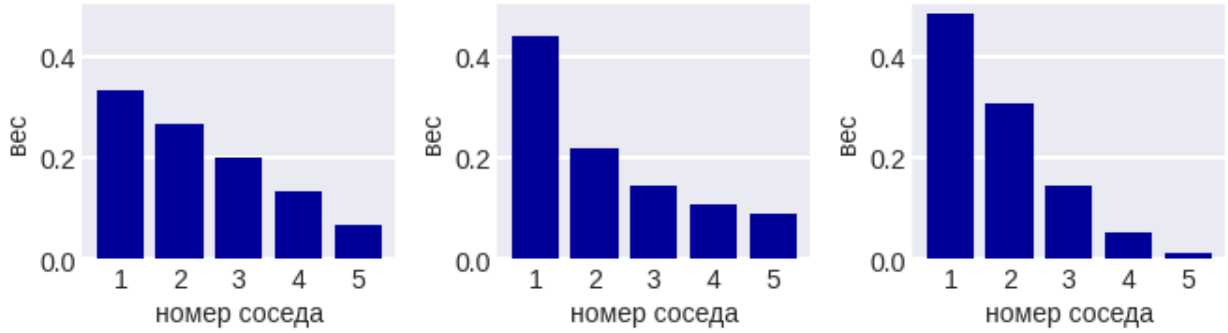

Рис. XX.8. Различные весовые схемы для пяти соседей.

Весовые обобщения метода kNN не просто получают классификации объектов, но и «достаточно разнообразные» оценки принадлежности к классам, т.к. в классическом варианте (XX.1) $b_j(x) \in \{i/k\}_{i=0}^{k}$, а в весовом (XX.2) значения $b_j(x)$ более разнообразны для всех $j \in \{1,2,\ldots,l\}$ [1].

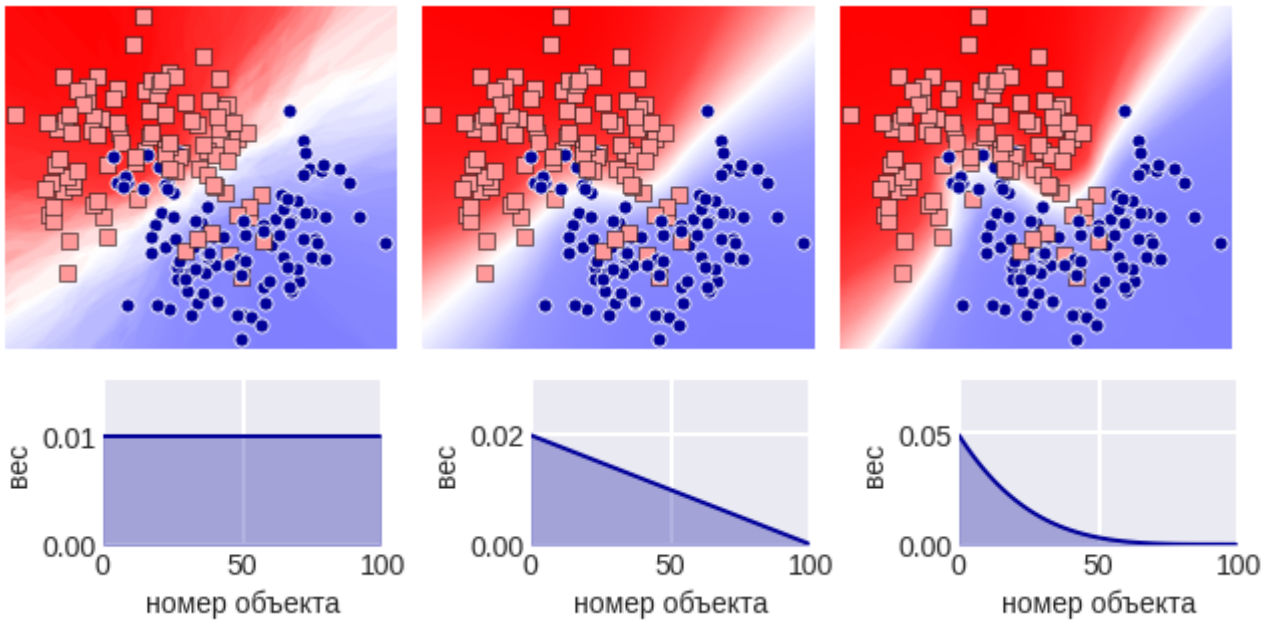

Рис. XX.9. Примеры весовых схем (справа – значения весов) и уверенность алгоритма в классификации (слева).

На рис. XX.9 показаны профили весов (нижний ряд) – т.е. значения весов первых 100 соседей и ответы алгоритма (верхний ряд) – более красные участки

[1] Это обеспечивает лучшее качество в задачах с функционалами качества типа AUC ROC (см. главу про критерии качества).

соответствую большей уверенности[1] в классе 0, более синие – в классе 1, белая полоса – разделяющей поверхности. Использование «более агрессивных» весовых схем (когда веса самых ближних объектов существенно превосходят остальные веса) геометрически похоже на уменьшение гиперпараметра k в классическом kNN.

Аналогично, возможны весовые обобщения kNN в задаче регрессии:

$$\frac{\sum_{t=1}^{k} w_t y(x_t)}{\sum_{t=1}^{k} w_t}$$

(аналог формулы (XX.2)), например, на рис. XX.10 показан эффект применения различных весовых схем в 5NN, менялась степень в (XX.1).

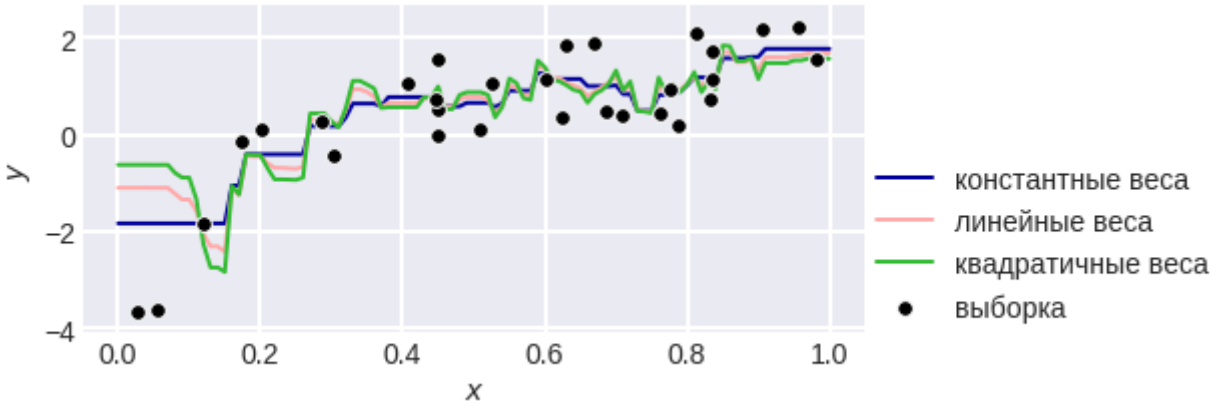

Рис. XX.10. Регрессионные кривые при различных весовых схемах.

Регрессия Надарая-Ватсона (Nadaraya-Watson Regression) это частный случай весовой kNN-регрессии при самой большой окрестности: $k = m$ и весах вида (XX.3) при константном[2] гиперпараметре $h(x) = h$, который называется **шириной ядра**, т.е.

$$w_i(x) = K\left(\frac{\rho(x, x_i)}{h}\right),$$

K – функция ядра (примеры ядер – в главе про математику в ML), чаще всего используется экспоненциальное ядро:

[1] оценке вероятности

[2] т.е. он не зависит от объекта x.

$$K(z) = \exp(-z).$$

Ответ алгоритма задаётся как взвешенное усреднение всех целевых значений:

$$a(x) = \frac{w_1(x)y_1 + \ldots + w_m(x)y_m}{w_1(x) + \ldots + w_m(x)}.$$

Смысл весов в формуле – чем ближе объект обучения, тем скорее ответ похож на его метку, см. рис. XX.11. Интересно, что эту формулу для ответа алгоритма можно вывести из решения задачи оптимизации:

$$\sum_{i=1}^{m} w_i(x)(a - y(x_i))^2 \to \min_a$$

(достаточно продифференцировать по a и приравнять производную к нулю), т.е. мы для объекта x ищем значение a, которое похоже на целевые значения. Но важность сходства со значением $y(x_i)$ задаётся весом $w_i(x)$, который тем больше, чем ближе объекты x и x_i друг к другу.

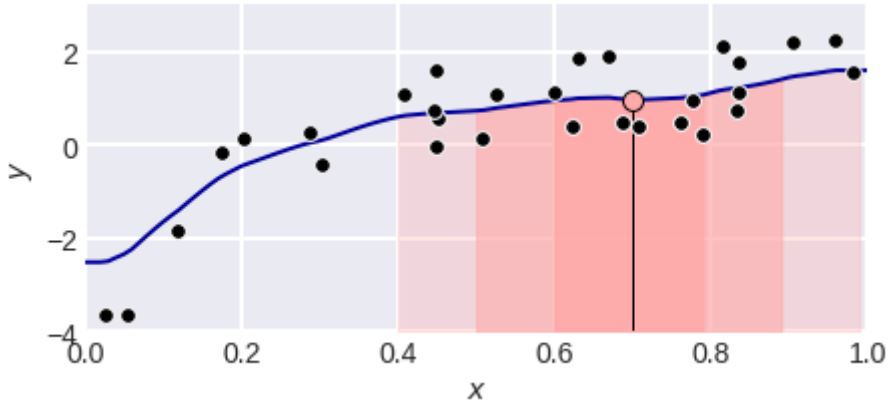

Рис. XX.11. Регрессия Надарая-Ватсона (интенсивностью розового показаны области близости к рассматриваемому объекту).

На рис. XX.12 изображены результаты применения регрессии Надарая-Ватсона при разной ширине ядра. Чем меньше ширина, тем лучше точки подгоняются под данные (точной подгонки здесь может не получиться).

Отметим, что регрессия Надарая-Ватсона не решает задачи экстраполяции, например, не улавливает линейный тренд (см. на рис. XX.12 на области левее левой точки выборки и правее правой), зато очень хороша для задач сглаживания сигналов (убирает мелкие колебания).

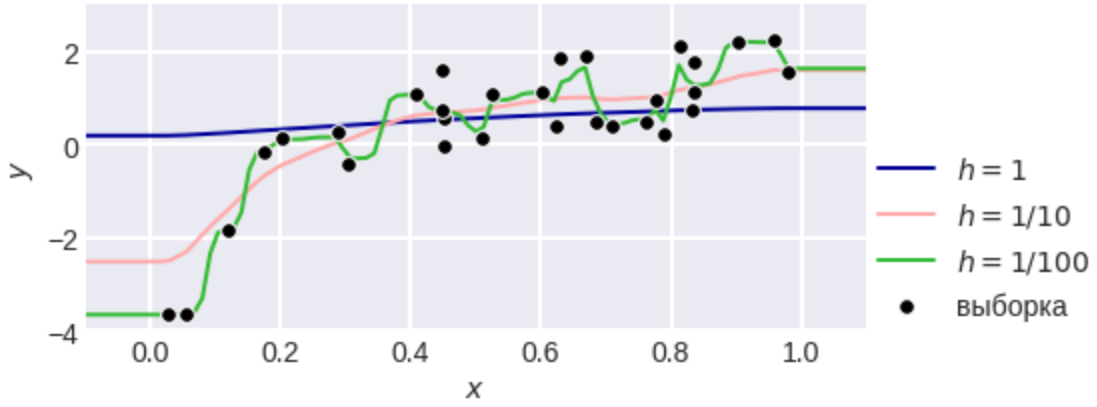

Рис. XX.12. Результаты регрессии Надарая-Ватсона при разной ширине ядра.

Проблема выбора метрики

В главе «Математика для машинного обучения» был обзор популярных метрик, отметим, что человек хорошо понимает геометрию эвклидовой метрики, но при использовании других метрик многое становится неинтуитивным. На рис. XX.13-14 показаны разделяющие поверхности поверхности метода 1NN при использовании L_1- и L_2-метрик. В белых областях на рис. XX.14 расстояние до ближайшего объекта класса 0 равно расстоянию до ближайшего объекта класса 1.

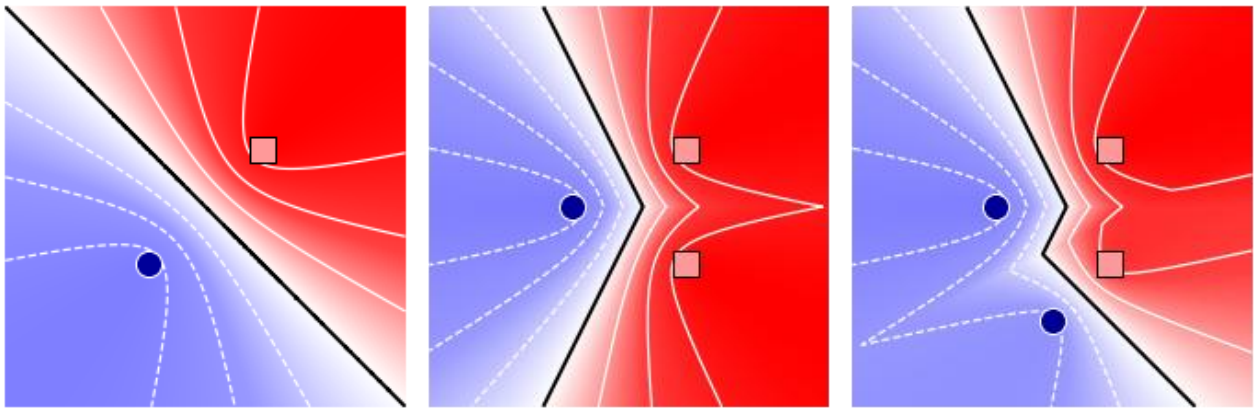

Рис. XX.13. Разделяющие поверхности метода 1NN при L_2 - метрике.

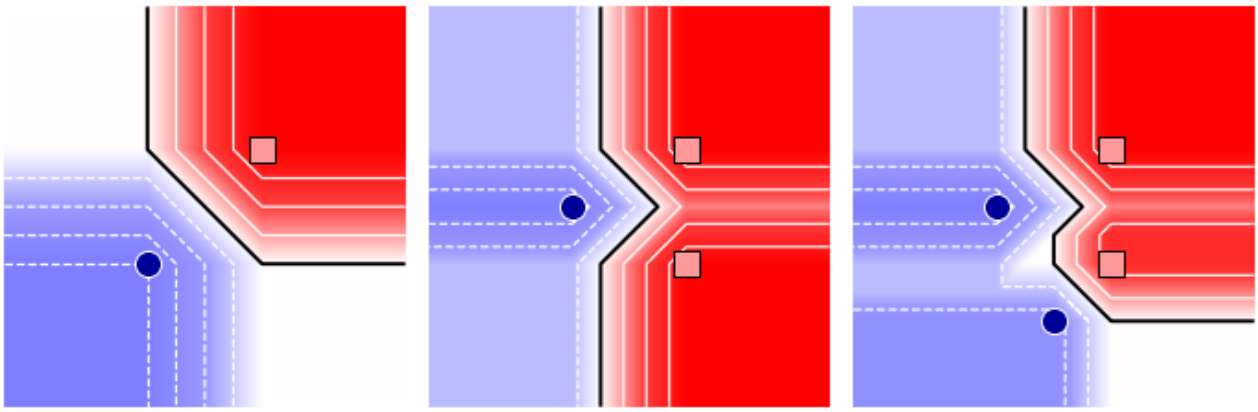

Рис. XX.14. Разделяющие поверхности метода 1NN при L_1 - метрике.

На практике надо понимать область применимости метрических алгоритмов. Особенно **они хороши в однородных признаковых пространствах** – когда все признаки в одном масштабе и имеют одинаковый смысл (см. примеры ниже). Часто, применяя их, используют нормировки (см. главу про предобработку данных) или используют метрику, которая является взвешенной суммой метрик в признаковых подпространствах (в которых имеет смысл вводить понятие расстояния между объектами). В подобных подпространствах выбор функции расстояния может быть, как ни странно, довольно прост. Можно также выбирать не метрику, а близость (например, косинусную меру сходства).

Выбор метрики существенно облегчается при наличии экспертных знаний о природе решаемой задачи и объектах.

В машинном обучении есть отдельный раздел **«Обучение метрики» (Metric Learning)**, в нём рассматривают параметризованное семейство метрик, например, метрику Махаланобиса

$$\rho(x,z)=\sqrt{(x-z)^{\mathrm{T}}\Sigma^{-1}(x-z)},$$

известную с точностью до матрицы Σ, которую обучают, т.е. настраивают так, чтобы качество решаемой задачи было максимальным. Иногда оптимизируют не качество решаемой задачи, а, например, подбирают матрицу Σ так, чтобы для каждого объекта расстояния до других объектов его класса было больше, чем до объектов чужих классов.

Рассмотрим задачу, представленную на рис. XX.15 (слева). Интуитивно хочется подправить метрику (объекты «размазаны» вдоль одного направления, а важно их различие по ортогональному направлению), давайте зададимся метрикой Махаланобиса с матрицей

$$\Sigma = \begin{bmatrix} \alpha & \beta \\ \beta & 1 \end{bmatrix}.$$

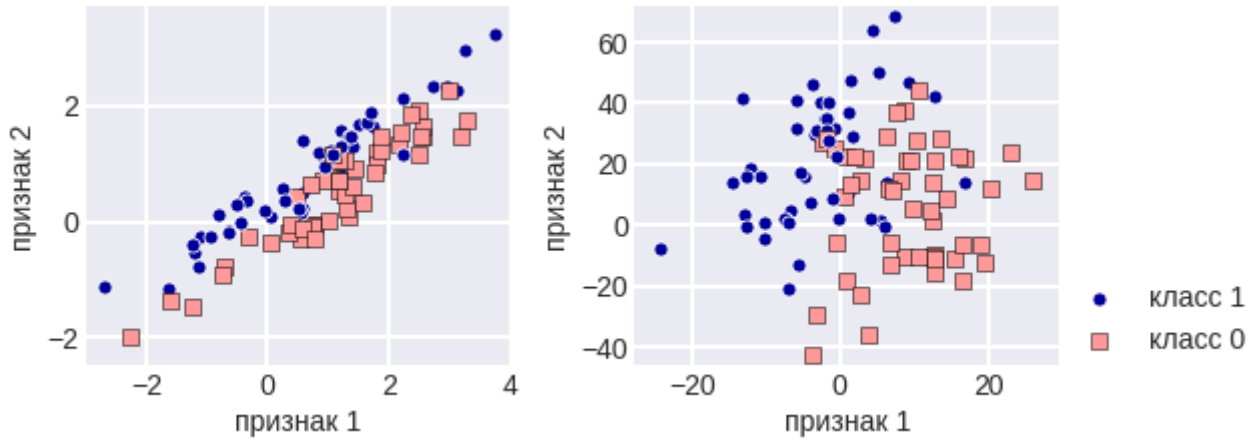

Рис. XX.15. Модельная задача бинарной классификации для выбора метрики (слева) и результат преобразования пространства после NCA (справа).

На рис. XX.16 показано, как качество зависит от параметров матрицы. Серая область соответствует значениям параметров, при которых матрица Σ не положительно определённая.

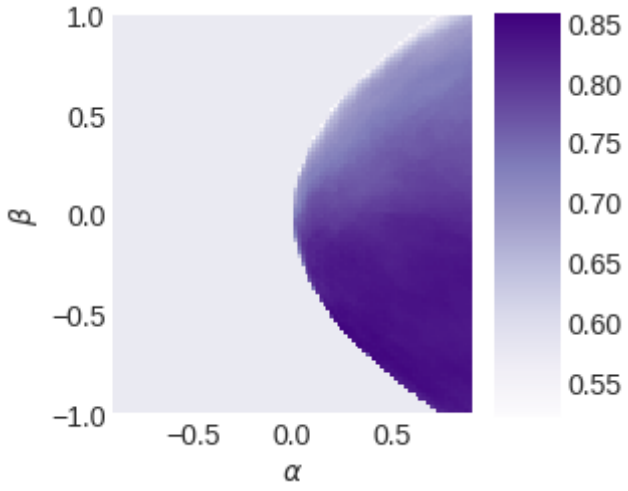

Рис. XX.16. Качество метода kNN с метрикой Махаланобиса при разных значениях параметров.

В методе **Neighborhood Components Analysis (NCA[1])** улучшается качество 1NN с евклидовым расстоянием, но не напрямую, а с помощью следующей процедуры: ищется преобразование $\varphi(x) = L(x-\mu): \mathbb{R}^n \to \mathbb{R}^n$, параметризованное матрицей $L \in \mathbb{R}^{n\times n}$ (вектор $\mu \in \mathbb{R}^n$ в дальнейших формулах сократится). Для этого вычисляются значения

$$p_{ij} = \frac{\exp(-\| Lx_i - Lx_j \|^2)}{\sum_{t\neq i}\exp(-\| Lx_i - Lx_t \|^2)},$$

а по ним $p_i = \sum_{j:y(x_j)=y(x_i)} p_{ij}$ (для i-го объекта суммируем по всем объектам его класса), решается задача оптимизации

$$\sum_{i=1}^{m} p_i \to \max.$$

На рис. XX.15 (справа) показано пространство объектов после применения найденного преобразования – кажется, что в новом пространстве уже логично использовать евклидову метрику.

Эффективный поиск ближайших соседей

Ниже кратко поговорим про быстрый и практичный поиск ближайших соседей. Соседей можно искать точно, а можно приближённо: с большой вероятностью находить точку, которая не сильно дальше, чем ближайшая. Понятно, что приближённые алгоритмы могут работать существенно быстрее и применяться для больших объёмов данных. Вместо подробного изложения современных алгоритмов поиска ближайших соседей, ограничимся идеями и используемыми структурами данных.

При поиске соседей в евклидовом пространстве популярны ***k*-мерные деревья (*k*D-деревья / *k*-d trees[2])**. На рис. XX.17 показано простейшее дерево глубины 1 – здесь значение первого признака сравнивается с порогом в вершине дерева. Геометрически это соответствует разделению пространства гиперплоскостью ортогональной первой оси координат – каждое полупространство соответствует

[1] Реализован в sklearn в классе sklearn.neighbors.NeighborhoodComponentsAnalysis.

[2] Bentley, J. L. (1975). «Multidimensional binary search trees used for associative searching». Communications of the ACM. 18 (9): 509–517.

листу дерева. Дерево можно продолжить строить рекурсивно: на рис. XX.18 показано дерево глубины 2. Для сбалансированности дерева можно по одной из координат выбирать медианное значение признака (тогда при разбиении области на подобласти в каждую попадёт около половины всех точек). Логично выбирать координату (признак) с максимальной дисперсией. При построении используется обучающая выборка, а при поиске ближайшего соседа тестового объекта, объект «спускается» по дереву – определяется область, в которой он находится (см. красный объект на XX.02). В этой области ищется ближайший сосед (т.е. перебираются не все объекты обучающей выборки, а только представители области) – часто так можно приближённо искать соседей, для точного нахождения соседа необходимо посмотреть в соседние области[1].

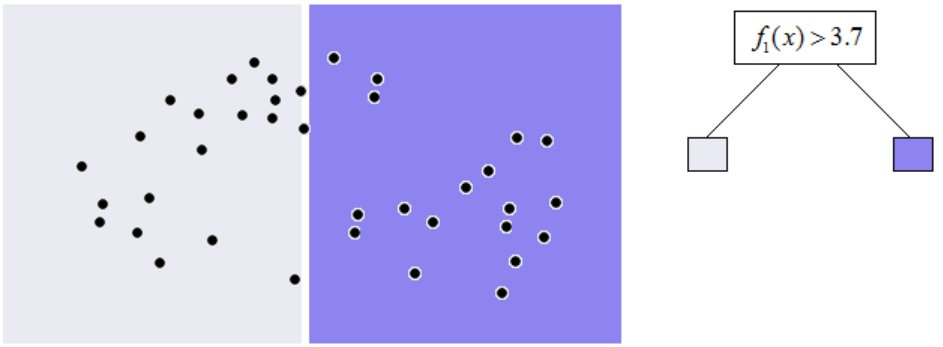

Рис. XX.17. *k*D-дерево глубины 1.

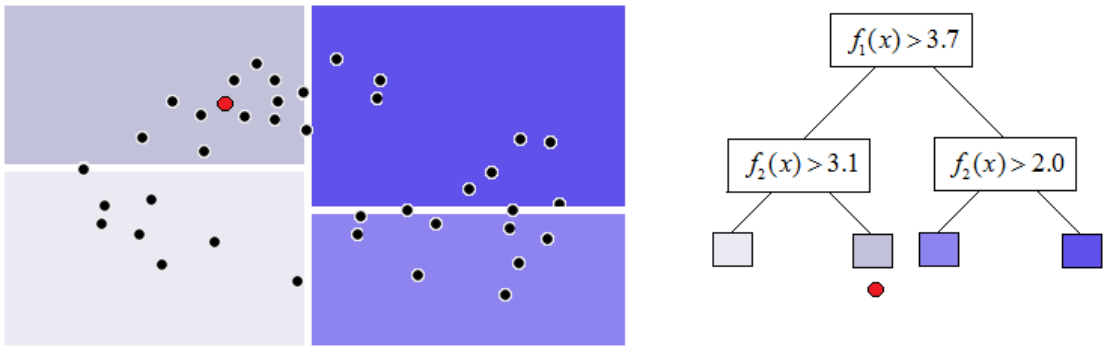

Рис. XX.18. *kD*-дерево глубины 2 и точка для поиска ближайших соседей.

Аналогично используются **деревья случайных проекций (Random Projection Trees[2])**, см. рис. XX.19, только в вершинах дерева сравнивают линейную комбинацию признака с порогом, поэтому области здесь имеют другую форму. У данного метода есть реализация **Annoy**[3] (в ней рекурсивно выбираются два объекта текущей области и область делится на две части с помощью гиперплоскости, которая лежит между объектами и ортогональна отрезку,

[1] см. упражнения.

[2] Sanjoy Dasgupta and Yoav Freund «Random Projection Trees and Low dimensional Manifolds». In 40th Anuual ACM Symposium on Theory of Computing, pages 537–546, 2008.

[3] https://pypi.org/project/annoy/

который их соединяет). Поскольку нет гарантии нахождения точного ближайшего соседа, можно построить несколько разных деревьев случайных проекций и все их использовать при поиске (перебирать объекты обучающей выборки из объединения областей, которые соответствуют листьям, в которые попал тестовый объект).

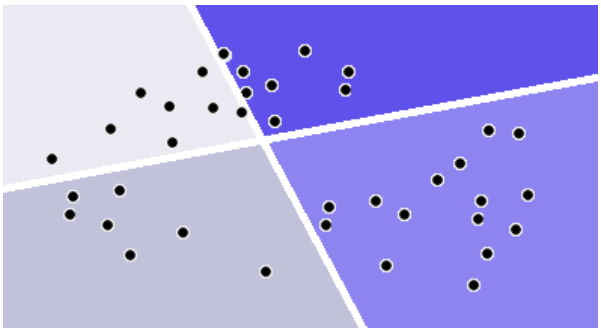

Рис. XX.19. Разбиение пространства деревьями случайных проекций.

В **шаровых деревьях (Ball Trees[1])** в вершинах сравниваются расстояния до двух точек, если объект ближе к первой – переходим в левый лист, иначе – в правый, см. рис. XX.20. Методы построения шаровых деревьев похожи на методы иерархической кластеризации (см. главу по кластеризации), например можно выбирать пару объектов с максимальным расстоянием с помощью них делать расщепление в дереве и спускаться на нижние уровни (там также ищем подобную пару объектов).

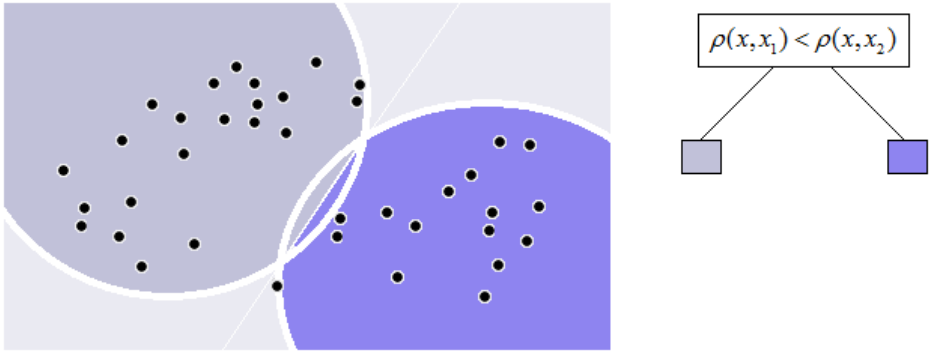

Рис. XX.20. Шаровые деревья.

При применении **граничных деревьев (Boundary Trees[2])** классификация объекта выглядит следующим образом. Пусть есть построенное дерево (его вершины – какие-то объекты обучающей выборки с метками) и тестовый объект (который нужно классифицировать), см. рис. XX.21. Спускаемся по

[1] Keinosuke Fukunaga and Patrenahalli M. Narendra. 1975. A branch and bound algorithm for computing k-nearest neighbors. IEEE Trans. Comput. 100, 7 (1975), 750–753.

[2] Mathy C. et al. The boundary forest algorithm for online supervised and unsupervised learning //Proceedings of the AAAI Conference on Artificial Intelligence. – 2015. – Т. 29. – №. 1.

дереву: смотрим расстояние до текущей вершины и её потомков (потомков может быть несколько). Если до какого-то потомка ближе – «спускаемся» (переходим в него), если нет, то считаем, что нашли соседа (текущую вершину). Например, на рис. XX.21 стартуем с корня – вершины 1, до вершины 2 ближе – переходим в неё, до вершины 3 ещё ближе – переходим в неё, у ней единственный потомок: вершина 4, но до неё дальше. Поэтому нашли соседа – вершину 3, классифицируем объект её меткой.

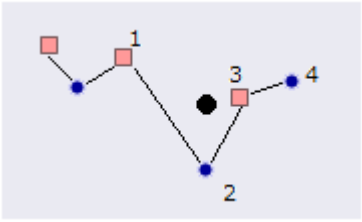

Рис. XX.21. Граничное дерево и объект для классификации (чёрный).

Строится граничное дерево по обучающей выборке: итерационно наращиваем дерево, перебирая объекты. Пусть построено дерево по k объектам обучения, берём $(k+1)$-й объект. Для него организуем поиск соседа, описанный выше. Если сосед имеет такую же метку, то переходим к следующему $(k+2)$-му объекту, иначе добавляем данный объект в дерево в виде потомка ближайшего объекта.

Известный принцип обучения: если алгоритм работает корректно – не трогаем, если нет – корректируем.

Название метода «граничные деревья» объясняется тем, что в такие деревья попадают, в основном, объекты на границе классов. На рис. XX.22 представлен результат эксперимента на модельной задаче: объекты являются точками плоскости, справа цветом показано, как они разделяются на классы, при построении дерева брались случайные точки, полученное граничное дерево показано слева[1].

Особенность большинства алгоритмов ML: больше внимания уделяется границам классов.

[1] Аналогичный метод можно использовать в задаче регрессии, также можно строить несколько граничных деревьев по случайным подвыборкам обучающей выборки и проводить по ним голосование (получаем ответы каждого дерева, итоговый ответ – самая частая метка из полученных), см. главу «Ансамбли».

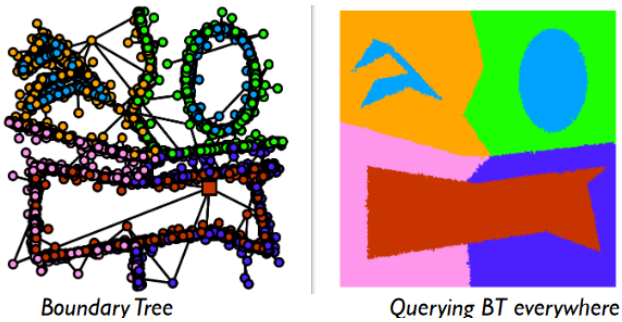

Рис. XX.22. Граничное дерево в модельной задаче: m = 100 000, в дереве $|V|$ = 2 220 [чужой рисунок].

При приближённом поиске ближайших объектов (особенно сложной природы) используется **хэширование с учётом близости (Locality-Sensitive Hashing – LSH[1])**, в котором близкие объекты имеют похожие хэши. Хэши выбирают короткие, поэтому их легко сравнивать. Для объекта ищут все объекты с таким же hash-значением (среди них ищем соседей), если таких объектов нет, то на все объекты (или используют другую hash-функцию).

Для эффективного поиска ближайшего соседа применяется также **метод квантования (Quantization)**. Сначала производят кластеризацию (разбиение выборки на группы похожих объектов, подробнее в главе про кластеризацию), для каждого кластера вычисляют центроид (центр масс кластера), для поиска ближайшего соседа, находят ближайший центроид и просматривают точки, которые лежат ближе к нему. Для этого используется т.н. «инвертированный индекс» (для каждого центроида хранят индексы ближайших точек, см. рис. XX.23).

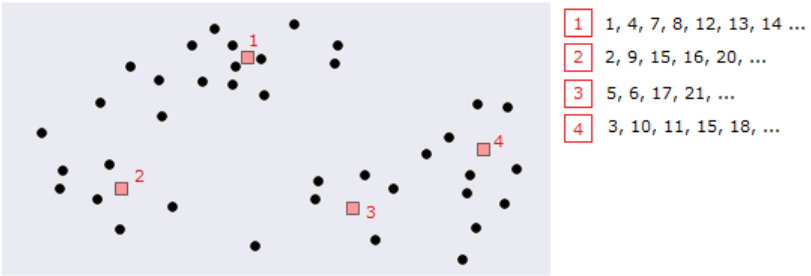

Рис. XX.23. Метод Quantization: выборка, нумерованные центроиды, индексы ближайших точек.

В методе **квантования произведения пространства (Product Quantization)** кластеризацию делают в разных признаковых подпространствах. Многомерное признаковое пространство делят на 2^k непересекающихся признаковых

[1] Rajaraman, A., Ullman, J. (2010). «Mining of Massive Datasets».

пространств[1], см. рис. XX.24 (там делится на 4 подпространства), в каждом подпространстве ищем 2^t центроидов. Каждый центроид можно записать числом от 0 до 2^{t-1}, т.е. с помощью t бит. Таким образом, на запись всего объекта (точнее центроидов, к которым он близок на соответствующих подпространствах) тратится $t+k$ бит. Теперь искать похожие объекты можно в низкоразметром пространстве: ищем объекты с похожим битовым кодом.

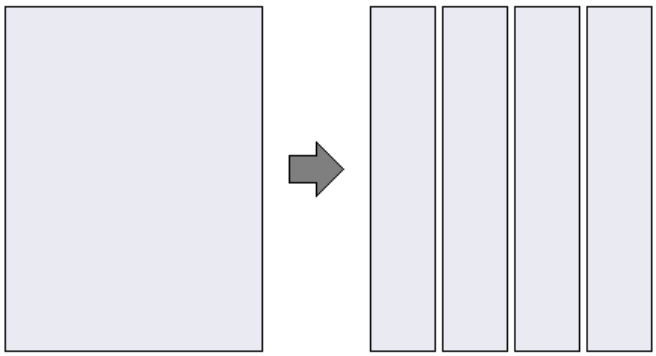

Рис. XX.24. Преобразование исходных данных.

На рис. XX.25 показана упрощённая версия квантования произведения: двупризнаковое пространство делится на два однопризнаковых, в каждом осуществляется кластеризация, в итоге пространство делится на 4 области: все точки одной области получают один битовой код. Виден основной недостаток метода: область с кодом (1,1) содержит небольшое число точек. Поэтому часто предварительно преобразуют пространство (с помощью поворота) или чередуют квантование (сначала ищут ближайший центроид, чтобы среди близких к нему точек искать соседей), повороты (правильно располагают соседей каждого центроида) и квантование произведения (для всех «повёрнутых» точек, близких к центроидам первого этапа[2]).

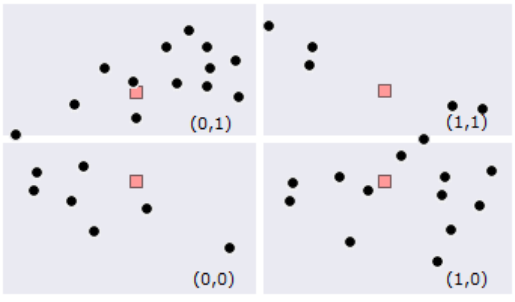

Рис. XX.25. Product quantization (упрощённая иллюстрация).

[1] Изложение по https://mccormickml.com/2017/10/13/product-quantizer-tutorial-part-1/

[2] Подробнее см., например, в Kalantidis Y., Avrithis Y. Locally optimized product quantization for approximate nearest neighbor search //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2014. – С. 2321-2328.

В заключение опишем оригинальный метод **Navigable Small World (NSW[1]),** в нём поиск очень быстрый, но нет гарантии точного нахождения соседа. Строится граф, вершины которого – объекты обучающей выборки. Рёбра в графе можно задавать по-разному: как в графе Делоне[2] (это гарантирует точное нахождение ближайшего соседа), но чаще используют граф k-соседства (каждую вершину соединяют с k ближайшими), также в граф обязательно добавляют случайные рёбра (чтобы граф стал «моделью малого мира[3]»).

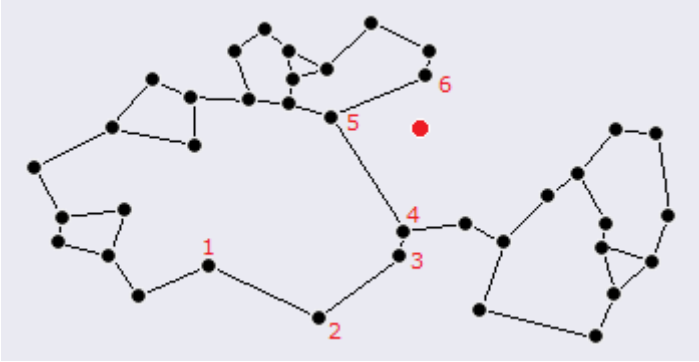

Рис. XX.26. Поиск точки по графу (красная точка – запрос, нумерация – перебор вершин графа).

Ближайшего соседа жадно ищем по графу, перебирая вершины. Начинаем с произвольной, перебираем соседей текущей вершины, считаем до них расстояния, переходим в ближайшую пока это возможно, см. рис. XX.26 (здесь начали с вершины 1 и далее перебираемые вершины пронумерованы, закончили перебор в вершине 6). Добавленные случайные рёбра позволяют быстро перемещаться по графу на большие расстояния при поиске.

В алгоритме **Hierarchical Navigable Small World (HNSW)** используют т.н. иерархические вложения: конфигурация точек прореживается (с некоторой вероятностью точка удаляется из неё), на прореженной конфигурации также строится граф. На рис. XX.27 справа показан следующий уровень иерархии, слева – последний (с совсем небольшим числом вершин). Поиск ближайшего соседа начинается сначала с небольшого графа (рис. XX.27, слева), после нахождения на текущем уровне иерархии переходим к следующему уровню иерархии (рис. XX.27, справа). Таким образом, нет необходимости по большому графу перебирать соседей вершин, большие графы используются для

[1] Malkov Y. A., Yashunin D. A. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs //IEEE transactions on pattern analysis and machine intelligence. – 2018. – Т. 42. – №. 4. – С. 824-836.

[2] https://ru.wikipedia.org/wiki/Триангуляция_Делоне

[3] В частности, для таких графов справедлива «теория нескольких рукопожатий»: для любой пары вершин существует путь на графе небольшой длины, который их соединяет.

уточнения соседа, когда мы его «почти нашли» (находимся уже в нужной области пространства). Описанный подход простой и эффективный, в том числе по памяти[1].

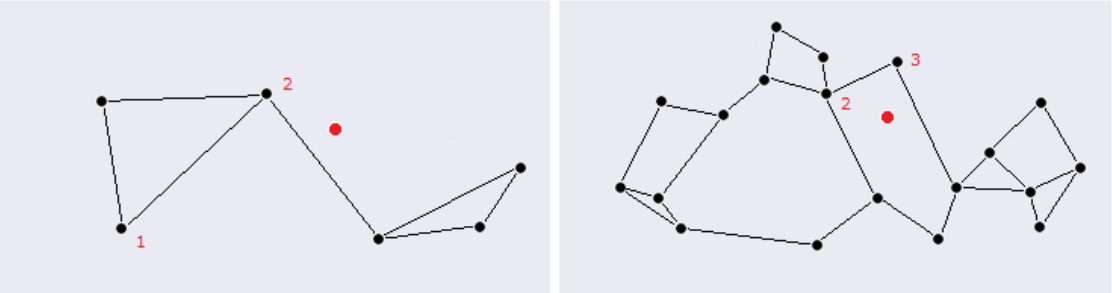

Рис. XX.27. Иерархический поиск точки по графу.

Приложения метрического подхода

Приведём несколько примеров использования метрического подхода на практике.

1. В одном из проектов потребовался **нечеткий матчинг таблиц**: были две таблицы, в каждой записаны данные о клиентах, и была гипотеза, что множества клиентов из двух таблиц достаточно сильно пересекаются, необходимо найти соответствия, подтвердить или опровергнуть гипотезу. Проблема заключалась в том, что в таблицах было много ошибок, записи делались в разные моменты времени (которые не отмечались). С приемлемым качеством задача решилась с помощью придумывания метрики между парами строк в разных таблицах стандартными средствами библиотеки scikit-learn. Например, при сравнении строки из одной таблицы со строкой из другой возраст должен был незначительно отличаться (сильные отличия штрафовались), а суммы последнего пополнения счёта могли отличаться больше.

2. Регрессия Надарая-Ватсона успешно использовалась в **задаче оценки скорости на дорогах**, правда не для самой оценки, а для предобработки данных. На рис. XX.28 показана скорость на отрезке дороги в разные моменты времени (чёрные точки – известные замеры), мы бы хотели получить график синего цвета, который показывает изменение скорости и согласован с данными. Заметим, что тут действует «двойное усредненение»:

[1] См. также https://habr.com/ru/company/mailru/blog/338360/

- по горизонтали (если известны скорости в два момента времени, а между ними не известна, например с 18:21 до 18:37, то график должен плавно проходить от одного замера к другому),

- по вертикали (если в один момент времени известно несколько скоростей – они могли сниматься с разных машин в одном потоке, то график должен проходить рядом со средней скоростью).

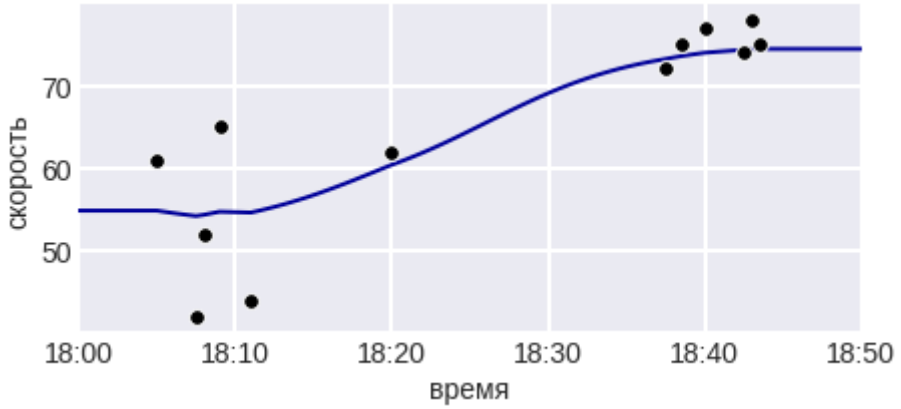

Рис. XX.28. Пример замеров скорости на участке дороги.

3. В **рекомендательных системах с холодным стартом (cold start)** также часто пригождаются метрики. В соревновании «VideoLectures.Net Recommender System Challenge[1]» (ECML/PKDD Discovery Challenge 2011) необходимо было рекомендовать лекции в режиме «холодного старта (cold start)»:

- рекомендовать надо из списка новинок, недавно загруженных на сайт (мы не знаем, насколько они интересны пользователям),

- рекомендовать надо неавторизованному пользователю, который начал смотреть лекцию (мы не знаем его данные и интересы, только описание просматриваемой лекции),

при этом известны описания всех лекций видеоресурса и статистика их просмотров другими пользователями. Решение-победитель использовало метрику[2]

$$\rho(\text{Lecture}_1, \text{Lecture}_2) =$$
$$= c_1 \cdot \rho_1(\text{Author}_1, \text{Author}_2) + c_2 \cdot \rho_2(\text{Title}_1, \text{Title}_2) + \ldots + c_r \cdot \rho_r(\text{Subject}_1, \text{Subject}_2),$$

[1] Дьяконов А.Г. Алгоритмы для рекомендательной системы: технология LENKOR // Бизнес-Информатика, 2012, №1(19), С. 32–39.

[2] На самом деле, использовалось более сложное алгебраическое выражение над элементарными метриками. Но сейчас для объяснения идеи метода это не принципиально.

которая являлось линейной комбинацией элементарных метрик на частях описания лекции: коллективах авторов, названиях и т.п. Элементарные метрики было придумать достаточно просто (например, для коллективов авторов можно использовать метрику Жаккара), а коэффициенты в линейной комбинации настраивались оптимизируя качество получаемых рекомендаций. Интересно, что большинство коэффициентов оказались нулевыми и решение зависело только от пяти элементарных метрик. Для рекомендации из списка новинок находились наиболее похожие по метрике лекции.

С помощью статистики просмотров элементарные метрики можно сделать нетривиальными. Рассмотрим, например метрику ρ_r на областях науки, которым посвящены лекции. Вместо тривиальной метрики

$$\rho_r(s_1,s_2)=\begin{cases}0, & s_1=s_2,\\ 1, & s_1\neq s_2,\end{cases}$$

хочется, чтобы метрика описывала интуитивное различие областей, например,

$$\rho_r(\text{«математика»},\text{«физика»})<\rho_r(\text{«химия»},\text{«история»}).$$

Для этого можно использовать идею: расстояние между областями тем меньше, чем чаще одни люди смотрят лекции из этих двух областей[1].

4. В **глубоком обучении (DL)** некоторые сети строят специальным образом, чтобы они превращали объекты (изображения, звуки и т.п.) в представления (вещественные векторы фиксированной длины) таким образом, чтобы близость в заданной метрике в этом пространстве имела смысл (схожие объекты имели небольшие расстояния между представлениями, а различающиеся – большие). Например, используют сиамские сети, см. рис. XX.29: по сути это одна сеть, которая изображение превращает в вектор, через неё пропускают пару изображений и требуют, чтобы для пар схожих изображений расстояние между векторами было небольшим, а для несхожих – большим[2]. Чаще всего сети «затачивают» на евклидову метрику или косинусное сходство. Под схожими изображениями понимаются изображения одного класса или разные версии одного (т.н. аугментированные версии – подвергшиеся незначительному изменению) или изображения одного объекта с разных ракурсов.

[1] см. упражнения.

[2] Для этого используют специальные функции ошибки, например Triplet Loss.

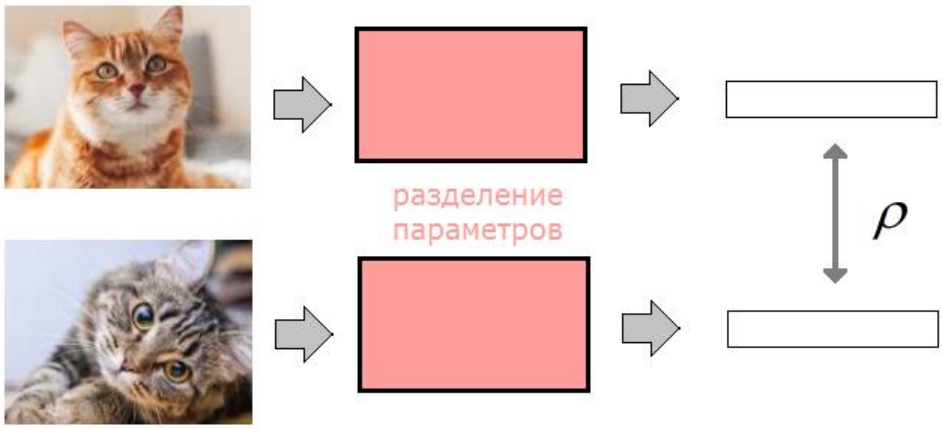

Рис. XX.29. Сиамская нейронная сеть.

Получение хороших представлений позволяет решать задачу поиска похожих изображений. Для набора изображений хранится набор представлений, если надо найти похожее изображение на заданное, оно переводится в представление и осуществляется поиск похожих представлений из набора, по которым выводятся соответствующие изображения.

5. Рассмотрим теперь **приложения метрического подхода в классификации текстов** на примере задачи «Large Scale Hierarchical Text Classification[1]». Здесь был использован взвешенный kNN, каждый текст задавался указанием ключевых слов, которые в него входят, и их количеством в тексте, текст мог принадлежать сразу нескольким классам. Поэтому обучающая выборка выглядела как представлено на рис. XX.30. Если задаться линейным порядком на множестве слов, то тексту соответствует вектор, в котором i-й элемент равен числу вхождения i-го слова. В таком векторе подавляющее большинство элементов равно нулю (он «разрежен»), т.к. в любом тексте небольшое число ключевых слов. Также каждому тексту сопоставлен бинарный вектор классификации, его длина которого равна числу классов, j-й элемент равен 1 тогда и только тогда, когда объект принадлежит j-му классу. Матрица объект-признак в этой задаче сильно разреженная, предварительно она нормировалась специальным образом[2]. Для классифицируемого объекта находились ближайшие k соседей (по косинусной мере сходства), их векторы классификации суммировались с весами (чем ближе объект, тем больше его вес). Так для q объектов тестовой выборки получалась вещественная матрица размера $q \times l$ (каждая строка – указанная сумма для соответствующего

[1] https://www.kaggle.com/c/lshtc

[2] Обычно используют tf-idf-нормировку.

объекта), после её бинаризации[1] получалась матрица-ответ (ij-й элемент равен 1, если i-й объект относим к j-му классу), см. рис. XX.31. Интересно, что в подобных задачах часто неплохое качество показывает простейший метод ближайшего центроида.

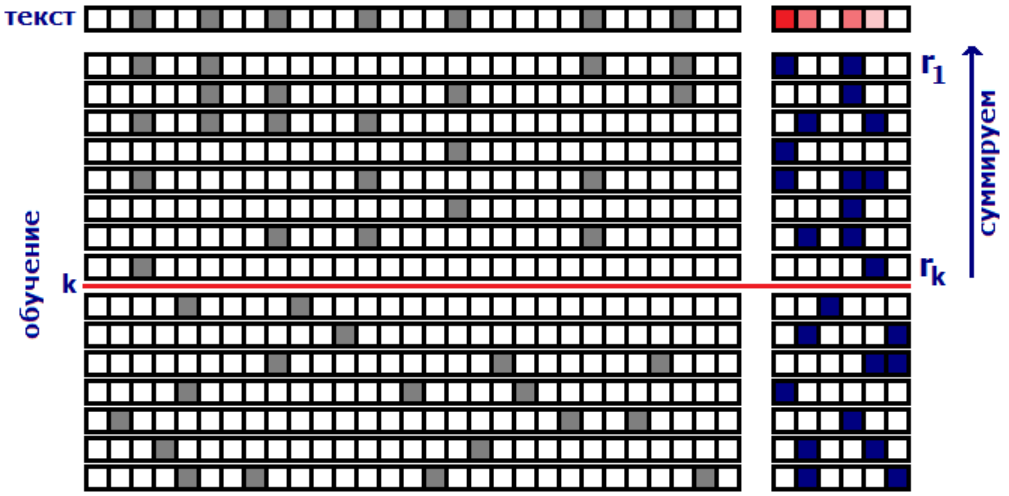

Рис. XX.30. Схематическое изображение весового kNN в задаче с текстами.

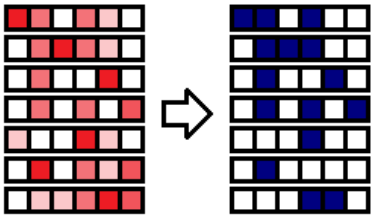

Рис. XX.31. Итоговая матрица оценок и её бинаризация.

6. Опишем ещё простой и любопытный метод, который занял третье место в соревновании **прогнозирования временных рядов** «NN5 Competition[2]». Пусть дан временной ряд $\tilde{f} = (f_1,\ldots, f_n)$, необходимо продолжить его, т.е. предсказать значения $(f_{n+1},\ldots, f_{n+t})$ для некоторого t. Рассмотрим предысторию прогноза: последний известный отрезок длины l (это параметр метода – «ширина окна») ряда $(f_{n-l+1},\ldots, f_n)$. Найдём похожие отрезки в истории ряда. Если найден похожий отрезок $(f_{k-l+1},\ldots, f_k)$, то искомое продолжение должно быть похоже (это гипотеза) на продолжение этого отрезка $(f_{k+1},\ldots, f_{k+l})$, см. рис. XX.32. На практике схожесть ищется с точностью до некоторого преобразования A:

$$\| A(f_{k-l+1},\ldots, f_k) - (f_{n-l+1},\ldots, f_n)\| \to \min_{k,\,A},$$

[1] Детали бинаризации пока пропустим, обычно при бинаризации сравнивают с фиксированным порогом, здесь применялась более хитрая техника.

[2] http://www.neural-forecasting-competition.com/NN5/results.htm

также разумно найти несколько похожих – ближайших соседей и усреднить их продолжения с учётом использованных преобразований

$$\sum_k c_k A_k(f_{k+1},\ldots,f_{k+l}),$$

здесь c_k – коэффициенты в усреднении, могут зависеть от степени похожести $(f_{k-l+1},\ldots,f_k)$ на предысторию $(f_{n-l+1},\ldots,f_n)$.

Рис. XX.32. Иллюстрация описанного метода прогнозирования значений ряда.

Преобразования можно использовать, например, линейные:

$$A(x_1,\ldots,x_l)=(a_1x_1+a_2,\ldots,a_1x_l+a_2).$$

Обратим внимание, что практически во всех приведённых прикладных задачах **признаки были однородные** (одинаковы по смыслу и в одной шкале), например в последней сравнивались подотрезки $(f_{k+1},\ldots,f_{k+l})$, здесь каждая компонента вектора – значение ряда. В задаче с текстами каждое значение признака – число вхождения соответствующего слова, в DL пространство «формировалось» под метрику. Не стоит вводить, например, евклидову метрику в признаковом пространстве с разнородными признаками (возраст, рост, вес, давление и т.п.)

Используйте метрики в однородных признаковых (под)пространствах.

Задачи и вопросы

1. Когда при использовании ближайшего центроида разделяющая поверхность не будет гиперплоскостью?

2. Можно ли идею ближайшего центроида использовать в задаче регрессии (мы рассматривали этот метод только для классификации)?

3. Как реализовать адаптивный метод ближайшего центроида (при постоянном пополнении обучающей выборки)?

4. Мы показали, сколько различных значений у (XX.1), а сколько их у (XX.2)?

5. Почему регрессию Надарая-Ватсона не всегда можно «настроить на данные» (добиться нулевой ошибки на обучающей выборке)?

6. В примере 3 практического приложения метрического подхода рассказывается о расстоянии между областями наук, которое можно вычислить по статистике просмотров. Предложите метод такого вычисления так, чтобы получалась метрика из заданного класса метрик, например, L_1-метрика.

7. На рис. XX.14 есть белые области равенства расстояний до ближайших объектов разных классов. Можно ли получить условие, при котором гарантируется отсутствие таких областей?

8. Как с помощью kD-деревьев производить точный поиск ближайшего соседа?

9. Почему при использовании графа k-соседства нет гарантии нахождения ближайшего соседа методом NSW? Обоснуйте использование графа Делоне для поиска ближайшего соседа.

Метрические алгоритмы: итоги

Резюмируем плюсы, минусы и особенности метрических алгоритмов:

+ не требуется признаковых описаний (достаточно уметь измерять расстояния / близости между объектами),

+ легко реализуемы,

+ решения хорошо интерпретируемы (можем предъявить похожие объекты с той же меткой, что и в ответе),

+ есть теоретические обоснования методов,

+ возможны различные модификации и обобщения методов[1], основанных на близости,

◊ алгоритм kNN ленивый – нет обучения (хотя при использовании эффективных методов поиска ближайших соседей в процедуре обучения могут создаваться специальные структуры данных),

◊ могут быть чувствительны к наличию выбросов / дисбалансу классов (но есть встроенные способы борьбы с этим, например выбор значения гиперпараметра k).

– медленная классификация (зависит от объёма обучения),

– метод kNN требуется хранение всей обучающей выборки, не эффективен на больших данных (по времени и памяти),

– требуется подбор метрики (возможно, нормировка признаков).

Спасибо за внимание к книге!
Замечания по содержанию, замеченные ошибки
и неточности можно написать в телеграм-чате
https://t.me/Dyakonovsbook

[1] см. также Bhatia N. et al. Survey of nearest neighbor techniques //arXiv preprint arXiv:1007.0085. – 2010; Cunningham P., Delany S. J. K-nearest neighbour classifiers-a tutorial //ACM computing surveys (CSUR). – 2021. – Т. 54. – №. 6. – С. 1-25.